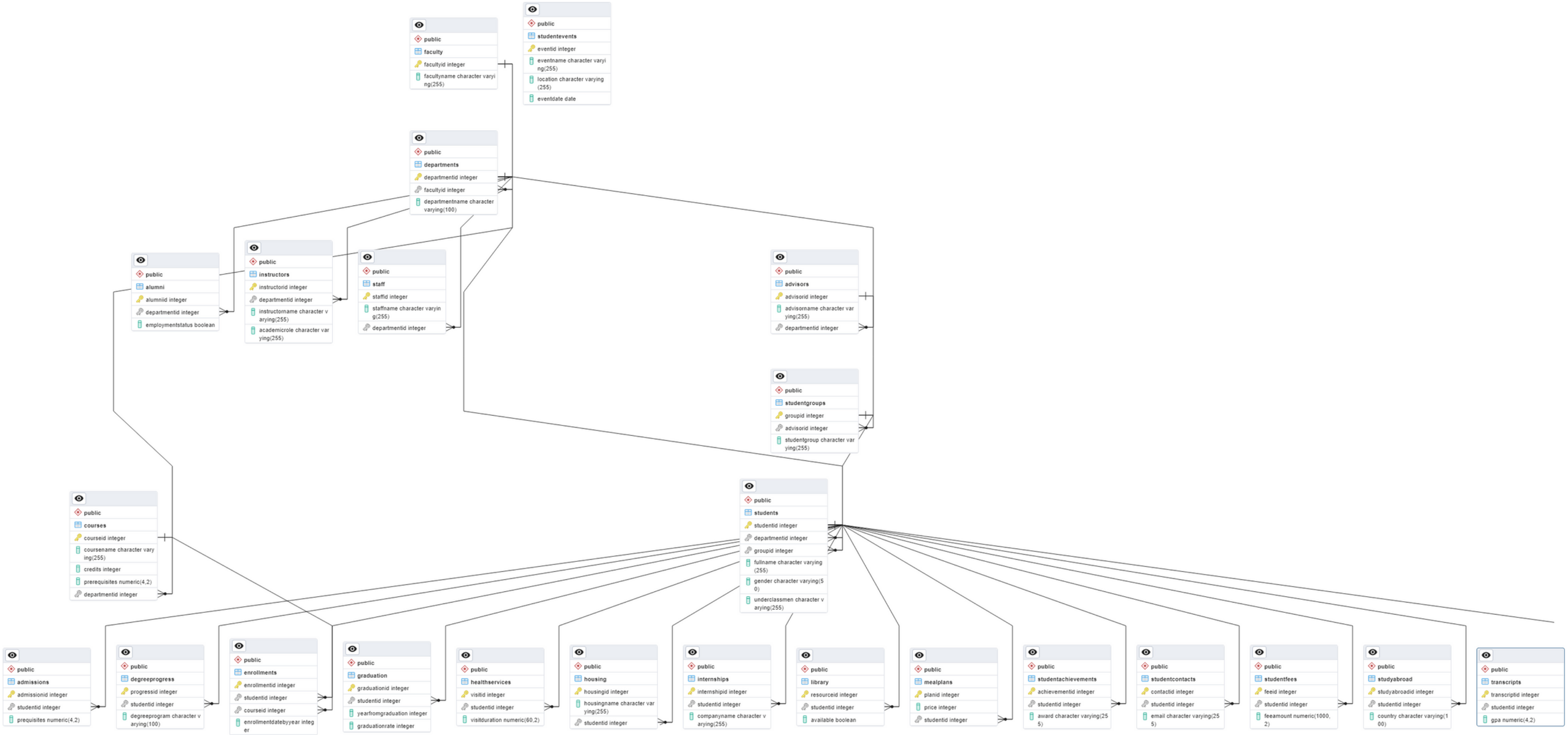
ERD diagram


public
faculty
facultyid integer
facultyname character varying(255)
public
studentevents
eventid integer
eventname character varying(255)
location character varying (255)
eventdate date
public
departments
departmentid integer
facultyid integer
departmentname character varying(100)
public
alumni
alumniid integer
departmentid integer
employmentstatus boolean
public
instructors
instructorid integer
departmentid integer
instructorname character varying(255)
academicrole character varying(255)
public
staff
staffid integer
staffname character varying(255)
departmentid integer
public
advisors
advisorid integer
advisorname character varying(255)
departmentid integer
public
studentgroups
groupid integer
advisorid integer
studentgroup character varying(255)
public
courses
courseid integer
coursename character varying(255)
credits integer
prerequisites numeric(4,2)
departmentid integer
public
students
studentid integer
departmentid integer
groupid integer
fullname character varying (255)
gender character varying(50)
underclassmen character varying(255)
public
admissions
admissionid integer
studentid integer
prequisites numeric(4,2)
public
degreeprogress
progressid integer
studentid integer
degreeprogram character varying(100)
public
enrollments
enrollmentid integer
studentid integer
courseid integer
enrollmentdatebyyear integer
public
graduation
graduationid integer
studentid integer
yearfromgraduation integer
graduationrate integer
public
healthservices
visitid integer
studentid integer
visitduration numeric(60,2)
public
housing
housingid integer
housingname character varying(255)
studentid integer
public
internships
internshipid integer
studentid integer
companyname character varying(255)
public
library
resourceid integer
studentid integer
available boolean
public
mealplans
planid integer
price integer
studentid integer
public
studentachievements
achievementid integer
studentid integer
award character varying(255)
public
studentcontacts
contactid integer
studentid integer
email character varying(255)
public
studentfees
feeid integer
studentid integer
feeamount numeric(1000,2)
public
studyabroad
studyabroadid integer
studentid integer
country character varying(100)
public
transcripts
transcriptid integer
studentid integer
gpa numeric(4,2)

1. List of Students and Their Enrollments

Description:

This code selects student's name and their enrollments and which faculty is their courses which they are enrolled

```
233  1.
234  SELECT s.FullName,f.FacultyName, e.EnrollmentDateByYear FROM
235  Students s
236  JOIN Enrollments e ON e.StudentID=e.StudentID
237  JOIN Departments d ON d.DepartmentID=s.DepartmentID
238  JOIN Faculty f ON f.FacultyID=d.FacultyID
239  ORDER BY E.EnrollmentDateByYear desc;
240
```

Data Output | Messages | Notifications

	fullname character varying (255)	facultyname character varying (255)	enrollmentdatebyyear integer
1	Gabey Rain	Science	2023
2	Marcelle De Malchar	Science	2023
3	Clevey Saffle	Science	2023
4	Orland Polleye	Engineering	2023
5	Budd Janous	Information Technology	2023
6	Devina Lanceter	Business	2023

Result:

It shows students and their enrollments

2.Retrieve a list of all students and the courses they are currently enrolled in, including course details.

Description:

Query selects student's name from Students and their enrollments and which courses is in their courses which they are enrolled with its Credits,Prerequisites by joining Enrollments,Courses tables

```
241  2.
242  SELECT s.FullName, c.CourseName AS EnrolledCourse, c.Credits, c.Prerequisites
243  FROM Students s
244  JOIN Enrollments e ON e.StudentID=s.StudentID
245  JOIN Courses c ON e.CourseID=c.CourseID;
246
247  3.
```

Data Output | Messages | Notifications

	fullname character varying (255)	enrolledcourse character varying (255)	credits integer	prerequisites numeric (4,2)
1	Iolanthe Sinderson	ZYO-469	3	2.16
2	Carlotta Town	ZYA-999	5	1.25
3	Kassia Becraft	OLD-614	9	2.92
4	Araldo Myrick	XHN-684	4	0.81
5	Wilmar Jeffree	CPO-751	8	2.52
6	Carlotta Town	WCK-204	7	3.96

Result:

The result returns a list of all students and the courses they are currently enrolled in with its Credits,Prerequisites details

3.Find the students who do not have assigned advisors.

Description:

This query retrieves StudentID and names of Students from Students table,whose GroupID is NULL(not assigned to group).Cause I connected advisors to StudentGroups table like in SDU, so that means if student does not have group,subsequently,he does not also have assigned advisors.

```
248  SELECT StudentID,FullName FROM
249  Students
250  WHERE GroupID is NULL;
251
```

Data Output | Messages | Notifications

	studentid [PK] integer	fullname character varying (255)
1	36	Staci Theze
2	54	Odelle Hughill
3	86	Ade Rabbitt
4	99	Florenza Timblett
5	103	Alair Puttrell
6	108	Hastie Condliffe

Result:

Lists the names of Students and their IDs who do not have advisors

4.Identify the student(s) with the highest GPA and their academic records.

Description:

Query retrieves students' name from Students and Fetches the GPA from the Transcripts table.
By joining Transcripts then it requests only those students whose GPA same as the max GPA in the Subquery cause in the questions marked the student(s) it means maximum GPA could be in two people or more.

```
252  4.
253  SELECT s.FullName, tr.GPA FROM
254  Students s
255  JOIN Transcripts tr ON tr.StudentID=s.StudentID
256  WHERE tr.GPA=(
257  SELECT MAX(GPA)
258  FROM
259  Transcripts
260  );
261
```

Data Output | Messages | Notifications

	fullname character varying (255)	gpa numeric (4,2)
1	Valdemar Lent	3.99

Result:

The output includes the full names of students (FullName) whose GPA matches the maximum GPA recorded in the Transcripts table.

5.Calculate the average GPA for students in each major.

Description:

This query calculates the average GPA (Grade Point Average) for each department by combining information from the Departments, Students, and Transcripts tables. It joins these tables based on the relationships between department IDs, student IDs, and their respective GPAs.

```
263  5.
264  SELECT d.DepartmentName, AVG(tr.GPA) AS AverageGPA FROM
265  Departments d
266  JOIN Students s ON s.DepartmentID=d.DepartmentID
267  JOIN Transcripts tr ON tr.StudentID=s.StudentID
268  GROUP BY DepartmentName;
269
```

Data Output | Messages | Notifications

	departmentname character varying (100)	averagegpa numeric
1	Information Systems	1.9877777777777778
2	Accounting	1.9675000000000000
3	Computer Science and Engineering	2.1216666666666667
4	Chemical Engineering	1.4457142857142857
5	Special Education	1.9750000000000000
6	Constitutional Law	2.9375000000000000

Result:

output of this query presents a table containing two columns:

- DepartmentName: Lists the names of different departments.
- AverageGPA: Displays the calculated average GPA for each department.

6.Determine which departments offer the most courses by counting the number of courses offered in each department.

Description:

The query identifies the top three departments offering the most courses by counting the number of courses available in each department. It accomplishes this by joining the Departments and Courses tables, counting the courses per department, and presenting the top three departments with the highest course counts.

```
271  SELECT d.DepartmentName, COUNT(*) AS CourseCount
272  FROM Departments d
273  JOIN Courses c ON d.DepartmentID = c.DepartmentID
274  GROUP BY d.DepartmentName
275  ORDER BY CourseCount DESC
276  LIMIT 3;
277
278  7.
```

Data Output | Messages | Notifications

	departmentname character varying (100)	coursecount bigint
1	Business Analytics	10
2	History	8
3	Software Engineering	8

Result:

Lists the department names and shows the count of courses offered within each department, highlighting the departments with the highest number of courses available

7.List faculty advisors along with the students they advise.

Description:

This query retrieves a list of faculty advisors along with the full names of students they advise. It achieves this by joining the Students, StudentGroups, and Advisors tables. The JOIN operations connect students to their respective groups and link those groups to their advisors. The results are sorted in ascending order based on the faculty advisors' names (AdvisorName).

```
279  SELECT a.AdvisorName, s.FullName
280  FROM Students s
281  JOIN StudentGroups sg ON sg.GroupID=s.GroupID
282  JOIN Advisors a ON a.AdvisorID=sg.AdvisorID
283  ORDER BY a.AdvisorName;
284
```

Data Output | Messages | Notifications

	advisorname character varying (255)	fullname character varying (255)
6	Adelheid Dyerson	Kerrin Kingescot
7	Adelheid Dyerson	Ashton Swinglehurst
8	Adelheid Dyerson	Matty Tortoise
9	Adrien Darque	Dosi Churms
10	Adrien Darque	Bren Drover
11	Adrien Darque	Jonas Taffee

Result:

Lists the names of faculty advisors and displays the full names of students advised by each respective faculty advisor. The list pairs each advisor with the students they advise, ordered alphabetically by advisor name

8.Find the student groups with most members and list the group names and member counts.

Description:

This query retrieves information about student groups, specifically displaying the GroupID, the StudentGroup name, and the count of members within each group. It achieves this by joining the StudentGroups table with the Students table based on the GroupID. The COUNT(*) function counts the number of students within each group. The results are grouped by GroupID and StudentGroup names, sorted in descending order by the count of members (MemberCount) within each group.

```
286  SELECT sg.GroupID,sg.StudentGroup, COUNT(*) AS MemberCount
287  FROM StudentGroups sg
288  JOIN Students s ON sg.GroupID = s.GroupID
289  GROUP BY sg.GroupID,sg.StudentGroup
290  ORDER BY MemberCount DESC;
291
292  9.--
```

Data Output | Messages | Notifications

	groupid [PK] integer	studentgroup character varying (255)	membercount bigint
1	59	Alpha Mu	9
2	89	Beta Upsilon	9
3	24	Omega	9
4	40	Sigma Pi	8
5	18	Sigma	8
6	11	Lambda	8

Result:

Displays the names and identifiers of the student groups,then it indicates the count of students present within each respective group. The list is ordered based on the number of members in descending order, showing the groups with the most members at the top.

9.Calculate the occupancy rate of the university's student housing facilities.

Description:

This query calculates the occupancy rate of the university's student housing by counting the number of Housing.StudentID which is occupied rooms and total rooms(H.HousingID itself) available, then computes the occupancy rate as a percentage by dividing the number of occupied rooms by the total rooms.

```
293  SELECT
294      COUNT(H.StudentID) AS OccupiedRooms,
295      COUNT(H.HousingID) AS TotalRooms,
296      (CAST(COUNT(H.StudentID) AS DECIMAL) / COUNT(H.HousingID))
297      * 100 AS OccupancyRate
298  FROM Housing H
299  FULL JOIN Students s ON s.StudentID = H.StudentID;
300
301  10.
```

Data Output | Messages | Notifications

	occupiedrooms bigint	totalrooms bigint	occupancyrate numeric
1	177	200	88.500000000000000000

Result:

- OccupiedRooms: The count of rooms currently occupied by students.
- TotalRooms: The total number of rooms available in the housing.
- OccupancyRate: The calculated occupancy rate as a percentage, representing the proportion of occupied rooms out of the total available rooms in the university's housing facilities.

10.Compute the average cost of meal plans for different student groups (e.g., freshmen, sophomores, etc.)

Description:

This query calculates the average price of meal plans for underclassmen students. It achieves this by joining the Students table with the MealPlans table based on the StudentID. The AVG(m.Price) function calculates the average price of meal plans for each category of underclassmen (Underclassmen column) and groups the results accordingly.

```
302  SELECT s.Underclassmen, AVG(m.Price) AS AveragePrice
303  FROM Students s
304  JOIN MealPlans m ON m.StudentID=s.StudentID
305  GROUP BY s.Underclassmen;
306
```

Data Output | Messages | Notifications

	underclassmen character varying (255)	averageprice numeric
1	senior	24.3333333333333333
2	sophomore	29.5555555555555556
3	junior	33.4444444444444444
4	freshman	29.0000000000000000

Result:

The output presents two columns:

- Underclassmen: Contains categories or types of underclassmen.
- AveragePrice: Displays the average price of meal plans for each category of underclassmen(such as freshmen, sophomores, etc).

11.Calculate the total tuition revenue generated by each academic department.

Description:

This query calculates the total tuition revenue generated for each department by summing the fee amounts paid by students. It accomplishes this by joining the Departments, Students in order to get into StudentFees tables and joins it. The SUM(f.FeeAmount) aggregates the fee amounts for each department, grouping the results by department name. And then sorts it by amount of fee.

```
308  SELECT d.DepartmentName, SUM(f.FeeAmount) AS TotalTuitionRevenue
309  FROM Departments d
310  JOIN Students s ON d.DepartmentID = s.DepartmentID
311  JOIN StudentFees f ON s.StudentID = f.StudentID
312  GROUP BY d.DepartmentName
313  ORDER BY TotalTuitionRevenue desc;
314
```

Data Output | Messages | Notifications

	departmentname character varying (100)	totaltuitionrevenue numeric
1	Computer Science	5320.84
2	History	4633.04
3	Linguistics	3047.40
4	Mathematics	2914.46
5	Literature	2598.95
6	Fine Arts	1925.84

Result:

Lists the names of different departments,then indicates the total revenue generated for each department by summing the fee amounts paid by students within that department. The departments are listed in descending order based on their total tuition revenue, showing which departments contribute the most to the overall revenue through student fees.

12.Find the number of available library resources and the number checked out by students.

Description:

This query retrieves information about the total number of resources in a library, the count of available resources, and the count of resources that are currently checked out. It achieves this by using aggregate functions and subqueries within a single query on the Library table like AvailableRecources is all recources(ResourceID) where it is available and so with already checkedouts

```
SELECT COUNT(ResourceID) AS TotalLibraryResources,
       (SELECT COUNT(*) FROM Library WHERE Available = TRUE) AS AvailableResources,
       (SELECT COUNT(*) FROM Library WHERE Available = FALSE) AS CheckedOutResources
FROM Library;
```

Data Output | Messages | Notifications

totallibraryresources bigint	availableresources bigint	checkedoutresources bigint
30	13	17

Result:

Indicates the total count of all resources in the library
Represents the count of resources that are currently available for use.
Displays the count of resources that are currently checked out and unavailable for use.

13.Calculate the number of student visits to health services and their average visit duration.

Description:

This query retrieves statistical information about visits to health services, including the total number of visits and the average duration of these visits in minutes. It operates on the HealthServices table, using aggregate functions to calculate these metrics. More clearly,it counts full VisitID which is the total number of visits and finds average duration by column VisitDuration

```
322  SELECT COUNT(VisitID) AS NumberOfVisits,
323         AVG(VisitDuration) AS AverageVisitDurationByMinutes
324  FROM HealthServices;
325
```

Data Output | Messages | Notifications

	numberofvisits bigint	averagevisitdurationbyminutes numeric
1	50	34.9600000000000000

Result:

Indicates the total count of visits to the health services.
Represents the average duration of visits in minutes.

14. List student achievements (awards, honors) and group them by the student's department.

Description:

This query retrieves information about students' achievements within their respective departments. It joins the Departments, Students, and StudentAchievements tables based on their relationships I made in order to find award for which deparment.

```
327  SELECT d.DepartmentName, s.FullName, sa.Award
328  FROM Departments d
329  JOIN Students s ON d.DepartmentID = s.DepartmentID
330  JOIN StudentAchievements sa ON s.StudentID = sa.StudentID
331  ORDER BY d.DepartmentName;
332
```

Data Output | Messages | Notifications

	departmentname character varying (100)	fullname character varying (255)	award character varying (255)
1	Accounting	Bren Drover	Journalist of the Year
2	Archaeology	Cori Turfes	Mathematician of the Year
3	Biology	Bendix Calderbank	Outstanding Athlete Award
4	Chemical Engineering	Adrian Greenhough	Photographer of the Year
5	Chemical Engineering	Gisella Spatarul	Historian of the Year
6	Computer Science	Budd Janous	Volunteer of the Year

Result:

The output displays the department names, full names of students, and their corresponding awards or achievements, ordered alphabetically by department names.

15. Determine the percentage of students who have participated in internships.

Description:

This query calculates statistics related to students and their participation in internships. It counts the total number of students, the count of unique students involved in internships, and computes the percentage of students engaged in internships out of the total student population. This calculation is performed based on data in the Students and Internships tables.

```
334  SELECT (SELECT COUNT(*) FROM Students) AS TotalStudents,
335  COUNT(DISTINCT StudentID) AS InternStudents,
336  (CAST(COUNT(StudentID) AS DECIMAL) /(SELECT COUNT(*) FROM Students))
337  * 100 AS percentage
338  FROM Internships;
339
```

Data Output | Messages | Notifications

	totalstudents bigint	internstudents bigint	percentage numeric
1	500	50	10.000000000000000000

Result:

Represents the total count of students in the Students table

Displays the count of distinct students involved in internships

: Shows the percentage of students engaged in internships out of the total student population.

16.Find the countries where students have studied abroad and the number of students in each country.

Description:

This query retrieves information about the number of students from each country participating in study abroad programs. It filters out NULL StudentIDs and groups the data by country using the StudyAbroad table.

```
341  SELECT Country, COUNT(StudentID) AS NumberOfStudents
342  FROM StudyAbroad
343  WHERE StudentID is NOT NULL
344  GROUP BY Country;
345
```

Data Output | Messages | Notifications

	country character varying (100)	numberofstudents bigint
1	France	4
2	United States	7
3	China	7
4	Netherlands	5
5	Australia	3
6	United Kingdom	4

Result:

- Lists the names of different countries from which students are participating in study abroad programs.
- : Displays the count of students from each respective country who are engaged in study abroad programs.

17.List the upcoming campus events and their details, sorted by date.

Description:

This query retrieves information about student events including the event name, location, and event date from the StudentEvents table. It sorts the results in ascending order based on the event date, listing events chronologically from the earliest to the most recent.

```
347  SELECT EventName, Location, EventDate
348  FROM StudentEvents
349  ORDER BY EventDate;
350
```

Data Output | Messages | Notifications

	eventname character varying (255)	location character varying (255)	eventdate date
1	Homecoming	Library	2022-12-06
2	Orientation	Cafeteria	2022-12-07
3	Hackathon	Gym	2022-12-08
4	Homecoming	Computer Lab	2023-01-21
5	Research Symposium	Computer Lab	2023-01-29
6	Orientation	Library	2023-02-02

Result:

Contains the names or titles of various student events.
Displays the locations or venues where these events take place.
Shows the dates of the events. The events are listed in ascending order based on their dates, starting from the earliest event date to the latest one

18. Determine which departments produce the most employed alumni.

Description:

This query retrieves information about the top five departments based on the count of employed alumni. It joins the Departments table with the Alumni table using a left join to link alumni to their respective departments. The query filters for alumni with an employment status of TRUE(if true then he is employed), counts the employed alumni for each department, and presents the results sorted in descending order by the number of employed alumni.

```
352  SELECT d.DepartmentName, COUNT(a.AlumniID) AS EmployedAlumnies
353  FROM Departments d
354  LEFT JOIN Alumni a ON d.DepartmentID = a.DepartmentID
355  WHERE a.EmploymentStatus = TRUE
356  GROUP BY d.DepartmentName
357  ORDER BY EmployedAlumnies DESC
358  LIMIT 5;
359
```

Data Output | Messages | Notifications

	departmentname character varying (100)	employedalumnies bigint
1	Visual Arts	10
2	Finance	10
3	Chemical Engineering	10
4	Economics	9
5	Cultural Studies	8

Result:

Names of the departments.
The count of alumni employed from each department.

19.Identify faculty members who have expertise in specific research areas, based on their academic records.

Description:

This query retrieves information about instructors who hold the academic role of 'PhD'(as they have expertise) within their departments. It joins the Instructors table with the Departments table and the Faculty table to link instructors to their respective departments and faculties. The query filters for instructors with an academic role of 'PhD' and displays the faculty name along with the names of instructors meeting this criteria. The results are ordered alphabetically by the faculty names.

```
361  SELECT f.FacultyName, i.InstructorName
362  FROM Instructors i
363  JOIN Departments d ON d.DepartmentID=i.DepartmentID
364  JOIN Faculty f ON d.FacultyID=f.FacultyID
365  WHERE i.AcademicRole = 'PhD'
366  ORDER BY f.FacultyName;
367
```

Data Output | Messages | Notifications

	facultyname character varying (255)	instructorname character varying (255)
1	Arts	Kiley Moultrie
2	Business	Lavinia Tompsett
3	Engineering	Fanchette Tomasoni
4	Law	Taffy Rubanenko
5	Law	Haze Mulles
6	Law	Nicki Gain

Result:

Names of the faculties to which the instructors belong.

Names of the instructors who have expertise in specific research areas within their respective faculties.

20.Analyze the historical enrollment data to identify trends in student enrollment over the past few years.

Description:

It counts the distinct number of students enrolled per year, grouping the data by the year of enrollment (EnrollmentDateByYear AS it is integer). The results are ordered in descending order based on the count of students enrolled in each respective year.

```
369  SELECT EnrollmentDateByYear,COUNT(DISTINCT StudentID) AS NumberOfStudents
370  FROM Enrollments
371  GROUP BY EnrollmentDateByYear
372  ORDER BY NumberOfStudents desc;
373
```

Data Output | Messages | Notifications

	enrollmentdatebyyear integer	numberofstudents bigint
1	2022	177
2	2020	166
3	2023	159
4	2019	158
5	2021	156

Result:

Represents the year of enrollment.
Then it indicates the count of distinct students enrolled in each respective year.

21.Verify if students enrolling in advanced courses meet the prerequisites by checking their transcript records.

Description:

This query retrieves information about students enrolled in courses for which they meet the prerequisites based on their transcript GPAs. It joins the Enrollments, Students, Courses, and Transcripts tables to link enrollment information, student details, course details, and transcript data. The query filters for enrollments where the student's GPA equals or exceeds the prerequisites required for the course.

```
375 SELECT s.FullName, c.CourseName
376 FROM Enrollments e
377 JOIN Students s ON s.StudentID=e.StudentID
378 JOIN Courses c ON e.CourseID=c.CourseID
379 JOIN Transcripts t ON e.StudentID = t.StudentID
380 WHERE t.GPA >= c.Prerequisites;
381
```

Data Output | Messages | Notifications

	fullname character varying (255)	coursename character varying (255)
1	Carlotta Town	ZYA-999
2	Araldo Myrick	XHN-684
3	Virgilio Pinchen	IME-595
4	Braden Cantillon	SHQ-699
5	Marjy Kasman	TUF-366
6	Roxie Parrington	SRU-362

Result:

Full names of the students enrolled in eligible courses.
Names of the courses in which these students are enrolled, meeting the prerequisite GPA requirements specified by the course.

22.List students with outstanding fees, including the total amount owed.

Description:

This query first of all gathers information about students who have outstanding fees(not zero). It performs a left join between the Students table and the StudentFees table based on the StudentID column. The query calculates the total fees paid by each student and filters the results to include only students who have a total fee amount greater than zero.

```
383  SELECT s.FullName, SUM(f.FeeAmount) AS TotalFee
384  FROM Students s
385  LEFT JOIN StudentFees f ON s.StudentID = f.StudentID
386  GROUP BY s.FullName
387  HAVING SUM(f.FeeAmount) > 0;
388
```

Data Output | Messages | Notifications

	fullname character varying (255)	totalfee numeric
1	Claresta Neeve	280.41
2	Stevana Loveman	155.53
3	Udall Cockell	977.64
4	Stanford Mc Trusty	1398.89
5	Layton Bengochea	612.99
6	Onfroi Westwood	189.24

Result:

Names of the students.

Total fees paid by each respective student, with results limited to students who have incurred fees greater than zero which guarantees us that it is outstanding.

23.Identify instructors who are teaching multiple courses in the same term and list the courses they are teaching.

Description:

Query retrieves information about instructors and the courses they teach, specifically focusing on instructors who are associated with departments offering more than five courses which are definitely multiple course.Then It joins the Instructors table with the Courses table based on the department IDs. The subquery identifies instructors who are associated with departments offering more than five courses.

```
390 SELECT i.InstructorName, c.CourseName
391 FROM Instructors i
392 JOIN Courses c ON i.DepartmentID = c.DepartmentID
393 WHERE i.InstructorName IN (
394 SELECT i.InstructorName
395 FROM Instructors i
396 JOIN Courses c ON i.DepartmentID = c.DepartmentID
397     GROUP BY I.InstructorName
398     HAVING COUNT(C.courseID) > 5
399 )
400 ORDER BY i.InstructorName, c.CourseName;
401
```

Data Output | Messages | Notifications

	instructorname character varying (255)	coursename character varying (255)
1	Betteann Salvati	JEC-363
2	Betteann Salvati	JHJ-842
3	Betteann Salvati	OMK-774
4	Betteann Salvati	ONR-080
5	Betteann Salvati	OQL-939
6	Betteann Salvati	PFL-391

Result:

Names of the instructors who are associated with departments offering more than five courses. Names of the courses taught by these instructors within the respective departments. The results are sorted alphabetically by the instructors' names, followed by the course names.

24.Calculate statistics on student diversity, such as the distribution of gender, ethnicity, or nationality.

Description:

First of all here I chose gender out of three distribution.
This query generates a count of students based on their genders. It utilizes the Students table, grouping the data by the Gender column. The COUNT(*) function tallies the number of occurrences of each gender category within the student population.

```
403  SELECT Gender, COUNT(*) AS Count
404  FROM Students
405  GROUP BY Gender;
406
```

Data Output | Messages | Notifications

	gender character varying (50)	count bigint
1	Female	242
2	Male	258

Result:

Represents the gender categories within the student data.
Displays the count of students belonging to each gender category.

25.Find the most popular combinations of courses (sets of courses taken together) among students.

Description:

This query retrieves information about students enrolled in multiple distinct courses simultaneously. It does so by joining the Enrollments table (twice) and the Students table to link student enrollments and student details. Additionally, it joins the Courses table (twice) to obtain course details for the enrolled courses.

The query compares and matches enrollments for a student in two different courses (Course1 and Course2) based on their StudentID. It ensures that the courses are distinct (e1.CourseID <> e2.CourseID and c1.CourseID <> c2.CourseID). The results are grouped by the full names of students along with pairs of distinct course names they are enrolled in simultaneously.

```
408  SELECT s.FullName, c1.CourseName AS Course1, c2.CourseName AS Course2
409  FROM Enrollments e1
410  JOIN Enrollments e2 ON e1.StudentID = e2.StudentID AND e1.CourseID <> e2.CourseID
411  JOIN Students s ON s.StudentID=e1.StudentID
412  JOIN Courses c1 ON c1.CourseID=e1.CourseID
413  JOIN Courses c2 ON c2.CourseID=e2.CourseID AND c1.CourseID <> c2.CourseID
414  GROUP BY s.FullName,Course1, Course2;
415
```

Data Output | Messages | Notifications

	fullname character varying (255)	course1 character varying (255)	course2 character varying (255)
1	Flor Bernardy	RMD-400	HYZ-041
2	Kassia Becraft	WCK-204	OLD-614
3	Mathilde Jemison	WCK-204	UAF-796
4	Dorey Szymonowicz	ESC-739	UEQ-155
5	Di Spore	GNG-700	RMD-400
6	Zolly Brearley	GYY-564	OQL-939

Result:

Names of the students.

Names of the first course in which the student is enrolled.

Names of the second distinct course in which the same student is simultaneously enrolled.

26.Compare the academic performance (GPA) of students based on their faculty advisors.

Description:

This query calculates the average GPA of students advised by each advisor. It achieves this by joining the Advisors, StudentGroups, Students, and Transcripts tables. The AVG(tr.GPA) function calculates the average GPA for each advisor based on the GPAs of the students they advise grouped by advisorName.

```
417  SELECT a.AdvisorName, AVG(tr.GPA) AS AverageGPA
418  FROM Advisors a
419  JOIN StudentGroups sg ON sg.AdvisorID=a.AdvisorID
420  JOIN Students s ON sg.GroupID = s.GroupID
421  JOIN Transcripts tr ON s.StudentID = tr.StudentID
422  GROUP BY a.AdvisorName;
423
424
```

Data Output | Messages | Notifications

	advisorname character varying (255)	averagegpa numeric
1	Pandora Benford	2.5550000000000000
2	Juliane Woodrough	3.0257142857142857
3	Vanny Vokins	2.5385714285714286
4	Hakim Dameisele	1.4250000000000000
5	Emylee Ghelerdini	2.4940000000000000
6	Kayley Leggott	1.9750000000000000

Result:

Names of the advisors.

The average GPA of the students advised by each respective advisor. Each row represents an advisor along with the average GPA of their advised stude

27.Identify student groups that have members from a wide range of majors, promoting interdisciplinary collaboration.

Description:

This query gathers information about student groups that represent more than three distinct majors. It joins the StudentGroups table with the Students table based on the GroupID column. Then, it counts the distinct department IDs represented within each student group. The HAVING clause filters the results to include only those student groups representing more than three distinct majors.

```
426  SELECT sg.StudentGroup, COUNT(DISTINCT s.DepartmentID) AS MajorsRepresented
427  FROM StudentGroups sg
428  JOIN Students s ON sg.GroupID = s.GroupID
429  GROUP BY sg.StudentGroup
430  HAVING COUNT(DISTINCT s.DepartmentID) > 3;
431
```

Data Output | Messages | Notifications

	studentgroup character varying (255)	majorsrepresented bigint
1	Alpha	9
2	Alpha Chi	7
3	Alpha Delta	4
4	Alpha Epsilon	4
5	Alpha Eta	6
6	Alpha Gamma	7

Result:

Names or identifiers of student groups.

The count of distinct majors represented within each student group that satisfies the condition (>3 distinct majors).

28.List courses with consistently high enrollment, helping with scheduling and resource allocation.

Description:

Here I chose course more than 7 enrollments. query retrieves courses with enrollment counts exceeding seven students. It joins the Courses table with the Enrollments table based on the CourseID column. Then, it counts the number of enrolled students for each course using COUNT(e.StudentID) and groups the data by the course name (c.CourseName). The HAVING clause filters the results to include only courses with an enrollment count greater than seven. Finally, the results are ordered in descending order based on the enrollment count.

```
433  SELECT c.CourseName, COUNT(e.StudentID) AS EnrollmentCount
434  FROM Courses c
435  JOIN Enrollments e ON c.CourseID = e.CourseID
436  GROUP BY c.CourseName
437  HAVING COUNT(e.StudentID) > 7
438  ORDER BY EnrollmentCount DESC
439
```

Data Output | Messages | Notifications

	coursename character varying (255)	enrollmentcount bigint
1	PUA-066	13
2	VPP-127	12
3	CXJ-331	12
4	WKE-513	10
5	HYD-296	9
6	MIA-642	9

Result:

Names of courses that have more than seven students enrolled.
The count of students enrolled in each respective course that meets the condition (>7 enrolled students), sorted in descending order of enrollment count.

29.Calculate the average time it takes students to graduate, considering their major and any changes in degree programs.

Description:

This query calculates the average year from graduation for students within each department. It joins the Departments, Students, and Graduation tables, associating students with their respective departments and their graduation years. The AVG(g.YearFromGraduation) function computes the average year from graduation for students within each department.

```
441 SELECT d.DepartmentName, AVG(g.YearFromGraduation) AS YearFromGraduatio
442 FROM Departments d
443 JOIN Students s ON s.DepartmentID=d.DepartmentID
444 JOIN Graduation g ON g.StudentID=s.StudentID
445 GROUP BY d.DepartmentName
446 ORDER BY YearFromGraduation;
447
```

Data Output | Messages | Notifications

	departmentname character varying (100)	yearfromgraduation numeric
1	Anthropology	1.7142857142857143
2	Environmental Science	1.7500000000000000
3	Management	1.7777777777777778
4	Art History	1.8000000000000000
5	Special Education	1.8750000000000000
6	Obstetrics and Gynecology	2.0000000000000000

Result:

Names of different departments.

The average year from graduation for students within each department. The results are ordered by the average year from graduation in ascending order.

30.Determine if students who complete internships have a higher graduation rate compared to those who do not.

Description:

This query calculates the average graduation rates for students with and without internships by using conditional aggregation with CASE statements. It separates students based on their presence or absence in the Internships table, then computes the average graduation rates accordingly.
LoL,my inserts were not considered here properly that is why avg grad rate with intern was lower than without xD

```
449 SELECT
450     AVG(CASE WHEN g.StudentID IN (SELECT StudentID FROM Internships)
451         THEN g.GraduationRate END) AS AvgGradRateWithInternship,
452     AVG(CASE WHEN g.StudentID NOT IN (SELECT StudentID FROM Internships)
453         THEN g.GraduationRate END) AS AvgGradRateWithoutInternship
454 FROM Graduation g;
455
```

Data Output | Messages | Notifications

	avggradratewithinternship numeric	avggradratewithoutinternship numeric
1	4.6400000000000000	5.5800000000000000

Result:

Represents the average graduation rate among students who have completed internships.
Indicates the average graduation rate among students who haven't participated in internships.